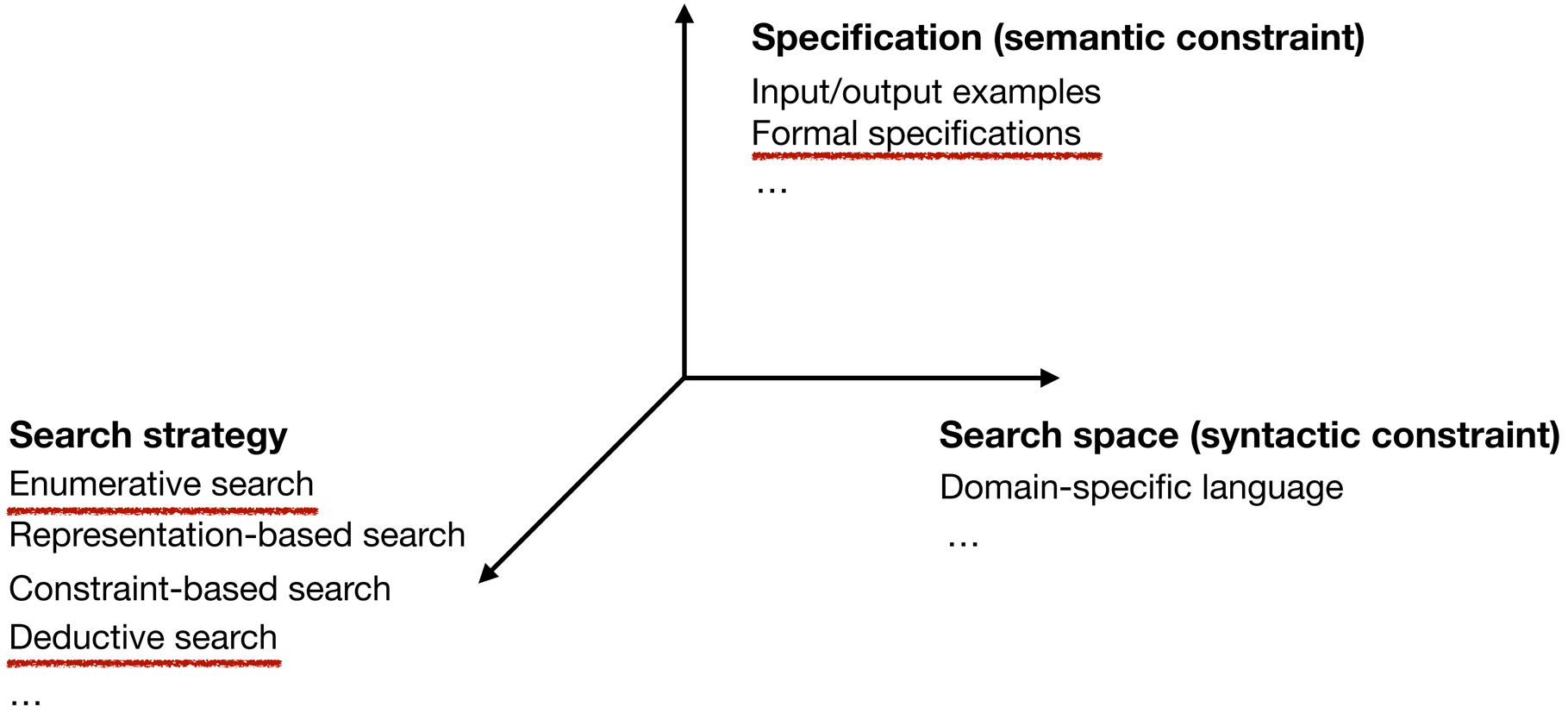
Program Reasoning

15. Functional Synthesis

Kihong Heo

KAIST

Dimensions in Program Synthesis



Functional Synthesis

$$\exists f . \forall x . P(f, x)$$

- Goal: find a function that satisfies a formal specification (i.e., logical formula)
  - Pre-condition: a predicate that all valid inputs to a function must satisfy
  - Post-condition: a post-condition that all outputs must satisfy
- In this lecture, two approaches based on
  - Enumerative search: enumerate all possible f until a desired one is found
  - Deductive search: repeatedly apply pre-defined transformation rules into the given spec until a desired one if found

Functional Synthesis via Inductive Synthesis

- Fact: we have learned many techniques for inductive synthesis
- Question: how can make a functional synthesizer using an inductive synthesizer?

(Hint)

Example

Specification

Find a function $f(x)$ where $\forall x, y\,.\, f(x,y) \geq x \wedge f(x,y) \geq y \wedge \big(f(x,y) = x \vee f(x,y) = y\big)$

Grammar

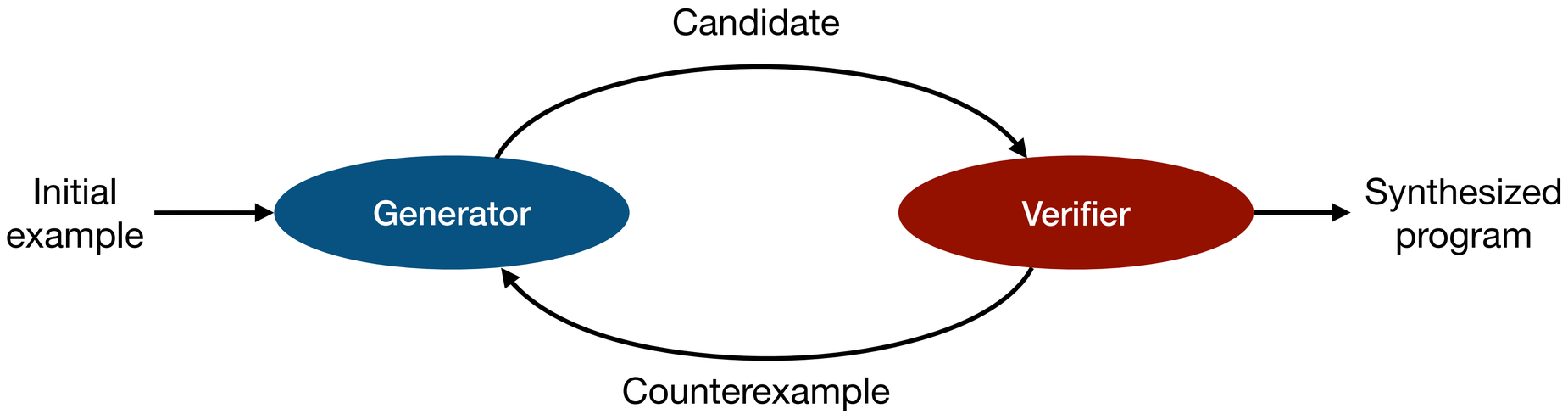
$$S \rightarrow x \mid y \mid S + S \mid S - S \mid \texttt{if}\ B\ S\ S$$
$$B \rightarrow S \leq S \mid S = S$$

Example

```
if(x >= y) x y
```

CEGIS

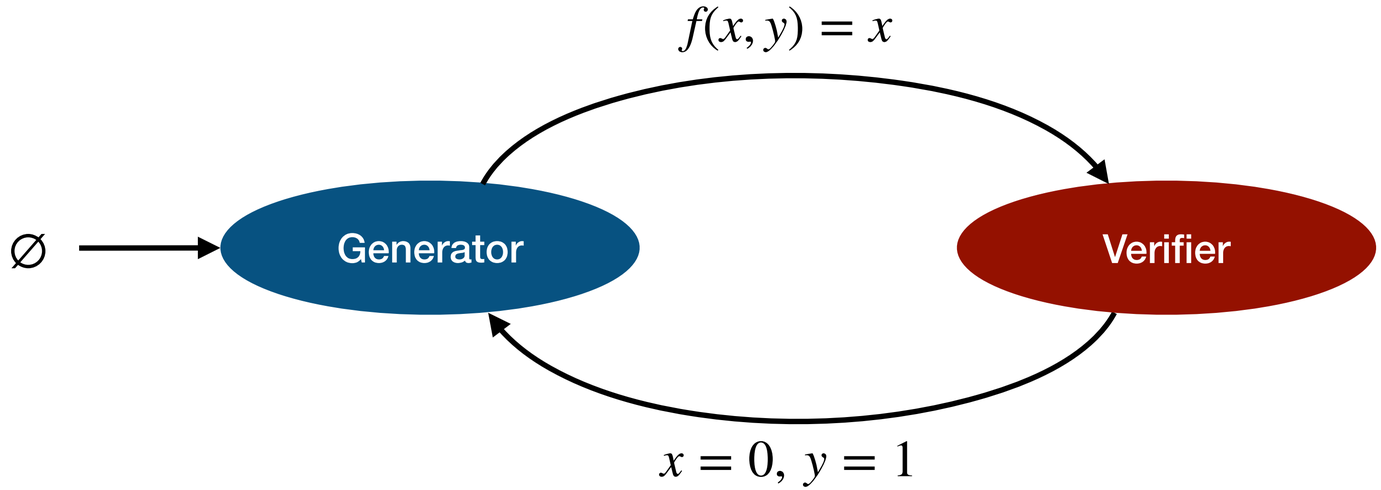
- CounterExample-Guided Inductive Synthesis
- A framework that enables us to use inductive synthesizers for functional synthesis
  - Generator: generate a candidate program (inductive synthesizer)
  - Verifier: check whether the candidate satisfies the specification (program verifier)



Example: CEGIS + Bottom-up Search

Specification

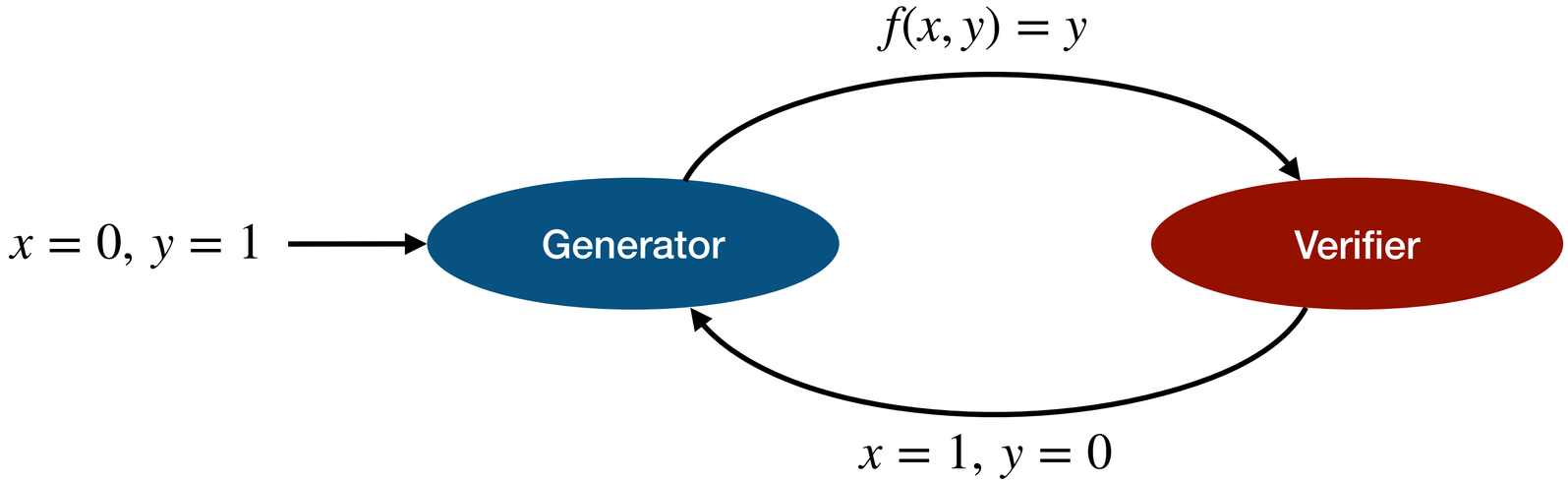
Find a function $f(x, y)$ where $\forall x, y\,.\, f(x, y) \geq x \wedge f(x, y) \geq y \wedge \big(f(x, y) = x \vee f(x, y) = y\big)$



Example: CEGIS + Bottom-up Search

Specification

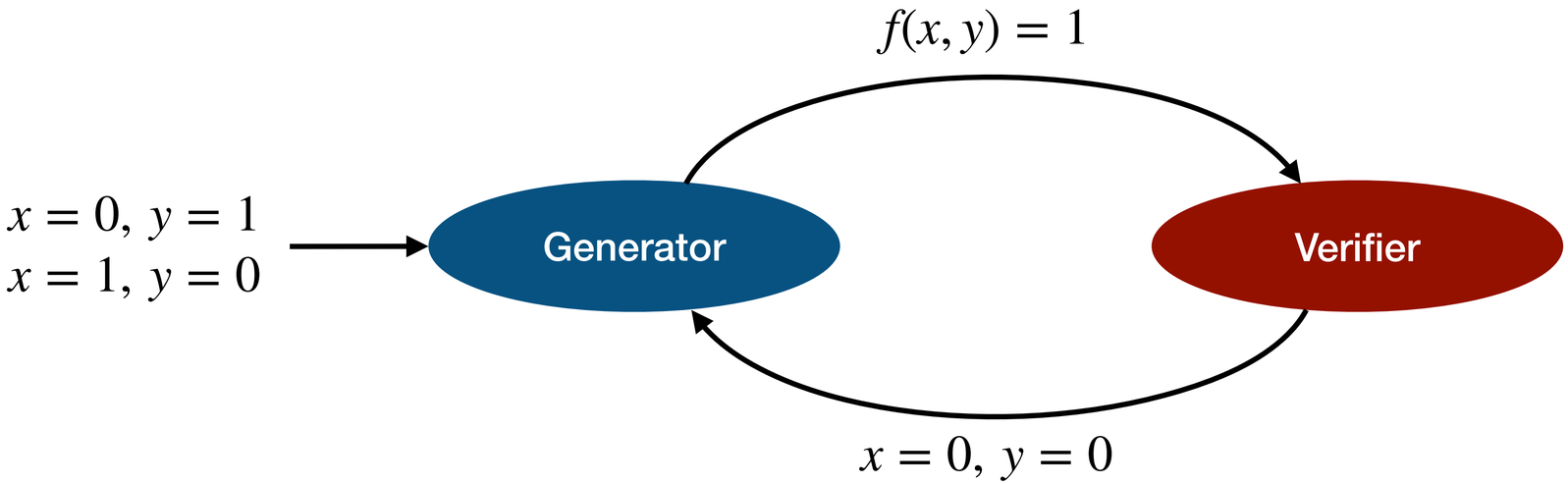
Find a function $f(x, y)$ where $\forall x, y\,.\, f(x, y) \geq x \wedge f(x, y) \geq y \wedge \big(f(x, y) = x \vee f(x, y) = y\big)$



Example: CEGIS + Bottom-up Search

Specification

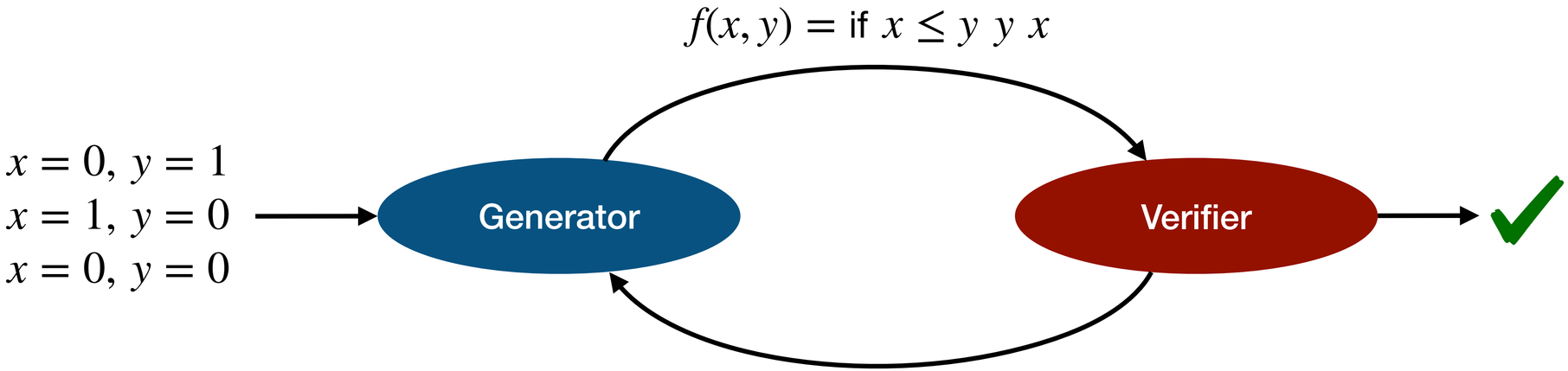
Find a function $f(x, y)$ where $\forall x, y\,.\, f(x, y) \geq x \wedge f(x, y) \geq y \wedge \big(f(x, y) = x \vee f(x, y) = y\big)$



Example: CEGIS + Bottom-up Search

Specification

Find a function $f(x, y)$ where $\forall x, y\,.\, f(x, y) \geq x \wedge f(x, y) \geq y \wedge \big(f(x, y) = x \vee f(x, y) = y\big)$



Benefits of CEGIS

- Generator and verifier are independent
  - Generator: enumeration-based, constraint-based, etc
  - Verifier: SMT, CHC, abstract interpretation, etc
- # CEGIS iterations: often small in practice
  - A candidate program which is correct w.r.t. a few examples is often a solution

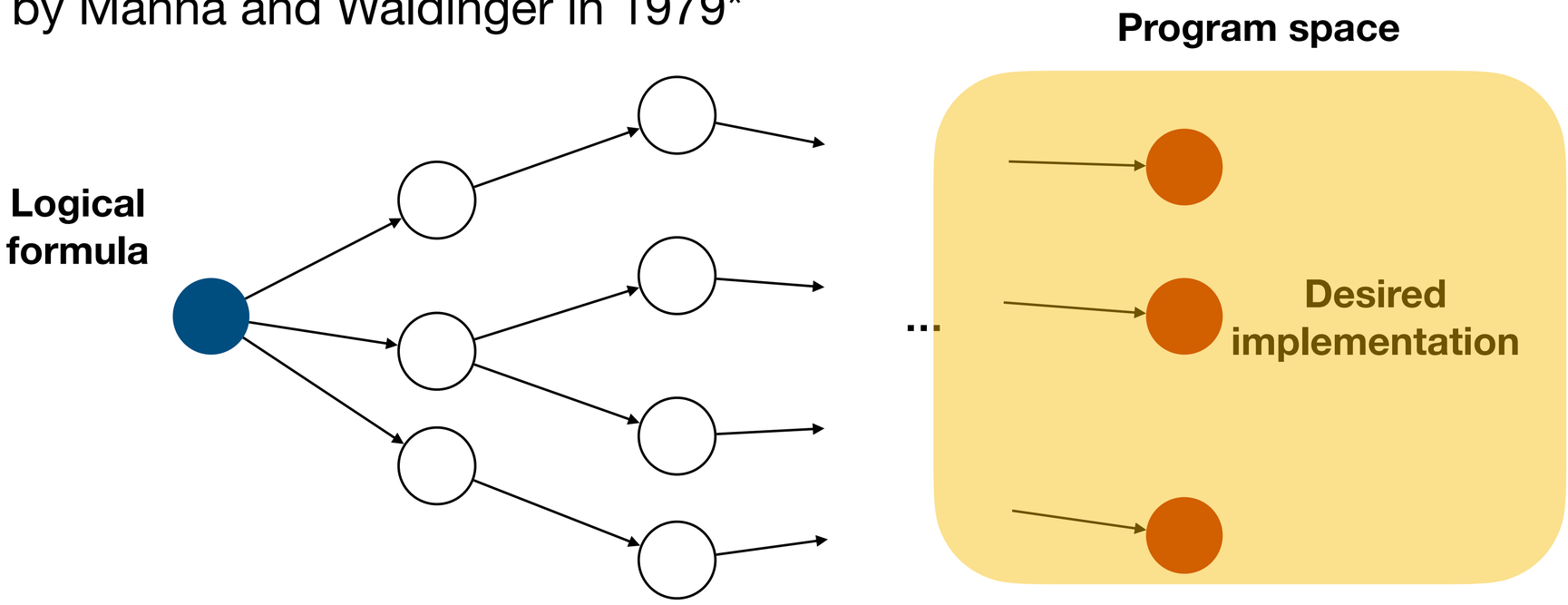
Functional Synthesis

$$\exists f . \forall x . P(f, x)$$

- Goal: find a function that satisfies a formal specification (i.e., logical formula)
  - Pre-condition: a predicate that all valid inputs to a function must satisfy
  - Post-condition: a post-condition that all outputs must satisfy
- In this lecture, two approaches based on
  - Enumerative search: enumerate all possible f until a desired one is found
  - Deductive search: repeatedly apply pre-defined transformation rules into the given spec until a desired one if found

Deductive Search

- Apply predefined transformation rules to derive a desired implementation
- Comparison to compilers: predefined order vs search
- Introduced by Manna and Waldinger in 1979*



*Z. Manna and R. Waldinger, *Synthesis: Dream → Programs*, IEEE Trans. of Soft. Eng.,1979

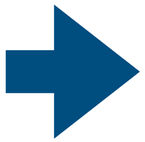
Example

- Synthesize a function $lesall(x, l)$ that determines whether x is less than all elements in list l

$\{x \in Number, l \in NumberList\}$

$\forall e \in l \,.\, x < e$

$\{\forall e \in l \,.\, x < e\}$



$\{x \in Number, l \in NumberList\}$

```
let lesall x l =
  if empty l then true
  else (x < head l) && lesall x (tail l)
```

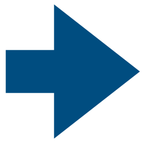
$\{\forall e \in l \,.\, x < e\}$

Transformation Rules

- Empty list: for any predicate ψ
  - $\{empty(l)\}\ \forall e \in l\,.\,\psi(e)\ \{Q\} \Longrightarrow \{empty(l)\}\ \texttt{true}\ \{Q\}$
- Conditional formation: for any predicate ψ
  - $\{P\}\ S\ \{Q\} \Longrightarrow \{P\}\ \texttt{if}\ \ \psi\ \ \texttt{then}\ S\ \text{else}\ S\ \{Q\}$
- Non-empty list
  - $\{P\}\,\forall e \in l\,.\,\psi(e)\ \{Q\} \Longrightarrow \{P\}\ \psi(head(l)) \wedge \forall e \in tail(l)\,.\,\psi(e)\ \{Q\}$
- Recursive calls: given a specification $\{P(x)\}\ f(x)\ \{Q(x)\}$, if $f(t)$ terminates
  - $\{P(t)\}\ Q(t)\ \{Q(t)\} \Longrightarrow \{P(t)\}\ f(t)\ \{Q(t)\}$

Example (1)

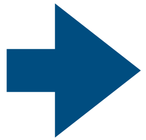
$\{x \in Number, l \in NumberList\}$
$\forall e \in l \,.\, x < e$
$\{\forall e \in l \,.\, x < e\}$



$\{x \in Number, l \in NumberList\}$
`if empty l then`
  $\{x \in Number, l \in NumberList, empty(l)\}$
  $\forall e \in l \,.\, x < e$
  $\{\forall e \in l \,.\, x < e\}$
`else`
  $\{x \in Number, l \in NumberList, \neg empty(l)\}$
  $\forall e \in l \,.\, x < e$
  $\{\forall e \in l \,.\, x < e\}$

Example (2)

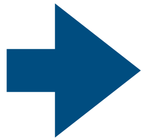
$\{x \in Number, l \in NumberList\}$
`if empty l then`
  $\{x \in Number, l \in NumberList, empty(l)\}$
  $\forall e \in l\,.\,x < e$
  $\{\forall e \in l\,.\,x < e\}$
`else`
  $\{x \in Number, l \in NumberList, \neg empty(l)\}$
  $\forall e \in l\,.\,x < e$
  $\{\forall e \in l\,.\,x < e\}$



$\{x \in Number, l \in NumberList\}$
`if empty l then`
  $\{x \in Number, l \in NumberList, empty(l)\}$
  `true`
  $\{\forall e \in l\,.\,x < e\}$
`else`
  $\{x \in Number, l \in NumberList, \neg empty(l)\}$
  $\forall e \in l\,.\,x < e$
  $\{\forall e \in l\,.\,x < e\}$

Example (3)

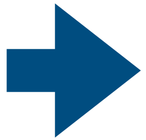
$\{x \in Number, l \in NumberList\}$
`if empty l then`
  $\{x \in Number, l \in NumberList, empty(l)\}$
  `true`
  $\{\forall e \in l \,.\, x < e\}$
`else`
  $\{x \in Number, l \in NumberList, \neg empty(l)\}$
  $\forall e \in l \,.\, x < e$
  $\{\forall e \in l \,.\, x < e\}$



$\{x \in Number, l \in NumberList\}$
`if empty l then`
  $\{x \in Number, l \in NumberList, empty(l)\}$
  `true`
  $\{\forall e \in l \,.\, x < e\}$
`else`
  $\{x \in Number, l \in NumberList, \neg empty(l)\}$
  `x < head l &&` $\forall e \in tail(l) \,.\, x < e$
  $\{\forall e \in l \,.\, x < e\}$

Example (4)

$\{x \in Number, l \in NumberList\}$
`if empty l then`
  $\{x \in Number, l \in NumberList, empty(l)\}$
  `true`
  $\{\forall e \in l \,.\, x < e\}$
`else`
  $\{x \in Number, l \in NumberList, \neg empty(l)\}$
  `x < head l &&` $\forall e \in tail(l) \,.\, x < e$
  $\{\forall e \in l \,.\, x < e\}$



$\{x \in Number, l \in NumberList\}$
`if empty l then`
  $\{x \in Number, l \in NumberList, empty(l)\}$
  `true`
  $\{\forall e \in l \,.\, x < e\}$
`else`
  $\{x \in Number, l \in NumberList, \neg empty(l)\}$
  `x < head l && lesall x (tail l)`
  $\{\forall e \in l \,.\, x < e\}$

Properties of Deductive Synthesis

- Correct by construction
  - Semantic-preserving transformation
  - No need to verify the correctness of a solution
- Usually domain specific
  - Need for pre-defined transformation rules

Summary

- Functional synthesis: find a function that satisfies a formal specification
- How to implement a *dream* as a *program*?
  - Enumerative search
  - Deductive search
- BTW, is logical formula really a dream?
  - Do you always come up with a logical formula before programming?
  - What other representations of dreams are useful in practice?